International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 9
May 2016

Secure Multi-Cloud framework by Homomorphic Encryption

[1]N.Swetha, [2]Prof. S Ramachandram

[1]Research Scholar, Dept. of CSE, Osmania University, Hyderabad

[2]Principal, Dept. of CSE, Osmania University, Hyderabad.

Abstract: *The possibility of Homomorphic encryption is to ensure data confidentiality while outsourcing data from cloud server or at cloud level, however with spare abilities to process over encoded information, seeking encrypted data information, and so on. A homomorphism is a property by which an issue in one algebraic framework can be changed over to an issue in another algebraic framework, be tackled and the arrangement later can likewise be changed over back effectively. In this way, homomorphism makes secure point out of calculation to an outsider attainable. Different liable encryption schemes have either multiplicative or additive substance Homomorphic property and are without further protest being used for individual applications. As such, a Fully Homomorphic Encryption (FHE) scheme which could play out any discretionary calculation over encoded data. In this paper, we propose a multi-cloud proposal of M conveyed servers to repartition the information (data) and to practically allow accomplishing a FHE.*

Keywords: *Multicloud .Homomorphic encryption, Fully Homomorphic Encryption.*

1. INTRODUCTION

Cloud computing has primitive services like IaaS, PaaS, and SaaS. SaaS plays a vital role where data storing and sharing. The vision of outsourcing an expanding measure of information storing and control to cloud supervisions raises numerous new security attentiveness toward people and organizations alike. The protection concerns can be agreeably tended to if clients encode the information they send to the cloud. In the event that the encryption scheme is Homomorphic, the cloud can in any case perform significant calculations on the information, despite the fact that it is encrypted

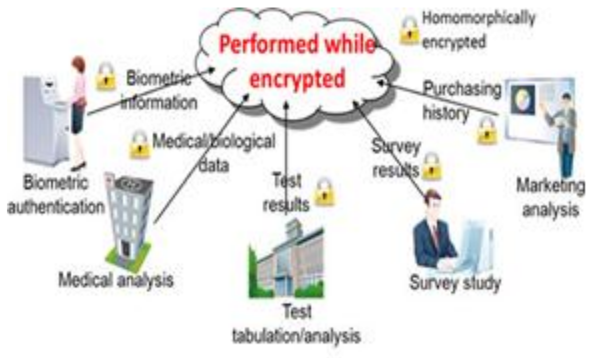
In any organization to perform some operations if they want to download confidential data from the cloud to a trusted computer and then send the encrypted results backed to the cloud, Cloud computing is infeasible for such business organizations. Encrypted data has previously been impossible to operate on without first decrypting them. Some encryption algorithms that permit arbitrary computation on encrypted data. For example, RSA is a multiplicatively homomorphic encryption algorithm where the decryption of the product of two encrypted data will be the product of the two plain data. On the other hand, RSA will not allow addition operation or the combination of additions and multiplications. Soon after, FHE has emerged [1] to carry out infinite chaining of algebraic operations in the cipher space, which means that a random number of additions and multiplications can be applied to encrypted operands. Unfortunately, all executions of FHE schemes proved that the performance is still slow for practical applications. In the last two years, solutions for fully holomorphic encryption schemes have been proposed and improved upon, but the problem faced with the efficiency.

In this paper we discuss the following: The Homomorphic encryption and interrelated definitions, its applications are defined in section I. In section II, we talk about the Homomorphic Scheme. In section III, we present some examples of partially holomorphic cryptosystems. In section IV, we propose a protected multi-cloud architecture for processing encrypted data. Section V deals with conclusion

2. BACKGROUND:

The expanding universality of cloud-based information and cell phones has prompted the rise of various new data services to address individuals' issues. In the meantime, there is an expanding consciousness of the issue of individual data getting to be open and of the should have the capacity to utilize individual information while keeping it private. Fujitsu has taken a proactive way to deal with protection assurance and has worked persistently on mechanical developments that will permit data to be utilized safely.

Encryption is a compelling approach to ensure information, despite the fact that, in most encryption strategies, information should be incidentally unscrambled so as to perform counts, for example, aggregates. This is tricky on the grounds that the information gets to be defenseless the minute it is decoded. Homomorphic encryption, be that as it may, considers estimations to be performed on information in an encoded state, making it a promising innovation for conveying new cloud services



(Figure 1). Data use with privacy protection in cloud service using Homomorphic encryption

3. HOMOMORPHIC ENCRYPTION

Homomorphic encryption is a type of encryption that permit calculations to be passed out on ciphertext, along these lines delivering an encrypted result which, when decoded, matches the aftereffect of operations do on the plaintext. Homomorphic encryption let the binding together of various presidencies without presenting the information to each of those supervisions. For instance, a chain of various managements from various organizations can ascertain 1) the request 2) the client exchange subtle elements 3) shipping, on an exchange without uncovering the decoded information to each of those services. Homomorphic encryption schemes are flexible by structure. This permits their necessity in distributed computing environment for guaranteeing the security of handled information. Alongside that the Homomorphic property of different cryptosystems can be utilized to make numerous other secure frameworks, for instance, secure voting frameworks, collision-resistant hash functions, private information retrieval schemes, and many more.

3.1 Practical Applications of Homomorphic Encryption:

Many approaches on Homomorphic encryption had been recognized very early. There are many applications which required a scheme that could work out homomorphically on encrypted data. But with the growing interest and tendency towards cloud computing has opened various possible application areas for Homomorphic Encryption. According to authors in [2] these applications can be majorly classified based on whether we expect privacy of data or circuit privacy or both. The categories are:

• Private Data, Public functions: like in Medical Applications.
•Private data, Private functions: like in Financial Applications.

The above mentioned applications assume single data (content) owner who encrypts the data and stores it on an untrusted cloud.

3.1.1 Electronic Voting: It is a unique case of allocation of calculation where one would like the election authorities to be able to calculate the votes and display the final results, but dislikes the idea that individual votes are first decrypted and afterwards tallied. In a voting system based on homomorphic encryption voters take turns incrementing an encrypted vote tally using a homomorphic operation. They are only allowed to increase the encrypted tally by 1 or by 0. Here 1 means indicating a vote for the candidate and 0 means indicating no vote for the candidate. In elections where each voter votes for one of N candidates, voters modify the encrypted tallies by adding an N-bit vector, where accurately one entry is 1 and the rest are all 0’s. They are not capable to alter the counters in any other way. Therefore, Homomorphic encryption is one of the solution for creating a “secret ballot” system online, wherever the votes will not reveal neither to anybody else except the voter.

3.2. A. Definition of a Homomorphic Encryption Scheme

A public-key encryption scheme S=(KeyGen, Encr, Decr) is homomorphic if for all N and all (pk,sk) output from KeyGen(k), it is possible to define groups T, E so that:The plaintext space T, and all ciphertexts output by Encrpkare elements of E.For any t1 , t2 ∈ T and e1 , e2 ∈ E with t1 = Decrsk (e1) and t2 = Decrsk (e2) it holds that:Decrsk (e1 * e2) = t1 * t2 Where the group operations ∗ are carried out in E and T, respectively.Similarly, a homomorphic cryptosystem is a PKS with the added property that there exists an efficient algorithm (Eval) to calculate an encryption of the sum or/and the product of two messages given the public key and the encryptions of the messages, but not the messages themselves.

Additionally, a fully Homomorphic scheme is capable to get output as a ciphertext that encrypts f (t1,...,tn), where f is any desired function, which of course must be calculated effectively. Information about t1,..., tn or f (t1,...tn), or any intermediate plaintext values will not leak. The inputs, outputs and intermediate values are always encrypted. Prior to take a closer look on fully homomorphic encryption schemes, we will need another important notion from information theory.

3.3 Circuits

Casually speaking, circuits are directed, acyclic graphs where nodes are called gates and edges are called wires. Depending on the nature of the circuit the input values are integers, boolean values, etc. and the matching gates are set operations and arithmetic operations or logic gates (AND, OR, NOR, NAND, ...). In order to calculate a function f, we express f as a circuit and topologically arrange its gates into levels which will be execute in sequence.

Example. Assume the function f outputs the expression

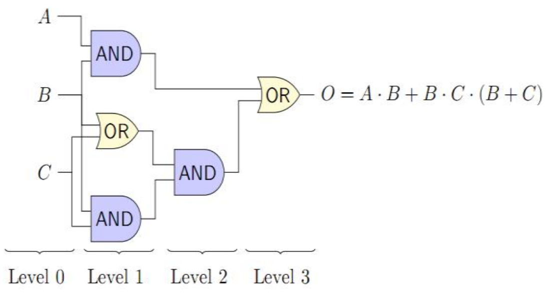


Fig. 1. Example for circuit representation

A•B+B•C•(B+C) on input (A,B,C). Then the following circuit represents the function f, with the logic gates AND and OR.

Two important complexity measures for circuits are size and depth.

The size of a circuit C is the number of its non-input gates. The depth of a circuit C is the length of its longest path, from an input gate to the output gate, of its underlying directed graph.

This yields to another definition of fully homomorphic encryption [8]:

ciphertexts Ψ = {c1 , ..., ct } where ci ← Encpk (mi), outputs

c← Evalpk (C, Ψ)

under pk.

To construct fully homomorphic encryption schemes we can also follow the second way. To know how this transformation works, we need the following definitions and corollaries.

Definition : A homomorphic encryption scheme E is said to be correct for a family CE of circuits if for any pair (sk, pk) output by KeyGenE (λ) any circuit C ∈ CE , any plaintext m1,...,mt , and any ciphertexts Ψ = c1, ...,ct

with ci ← Encpk (mi), it is the case that:

If c ← EvalE (pk, C, Ψ), then DecE (sk, c) → C(m1, ...,mt) Except with negligible probability over the random coins in EvalE .

Definition: A homomorphic encryption scheme E is compact, if there is a polynomial f so that, for every value of the security parameter λ, E's decryption algorithm can be expressed as a circuit DE of size at most f (λ).A homomorphic encryption scheme E efficiently evaluates circuits in CE if E is compact and also correct for circuits in CE.

Corollary: A homomorphic encryption scheme E is fully homomorphic if it compactly evaluates all circuits.

This requirement is considered to be approximately too strong for practical purpose, therefore it uses a certain relaxation to comprise leveled schemes, which only estimate circuits of depth up to some d, and whose public key length may be poly(d).

Definition: (leveled fully homomorphic). A family of homomorphic encryption schemes {E(d) : d ∈ Z+ } is said leveled fully homomorphic if, for all d ∈ Z+ , it all uses the same decryption circuit, E (d) compactly evaluates all circuits of depth at most d (that use some specified set of gates), and the computational complexity of E (d) 's algorithms is polynomial in λ, d, and (in the case of EvalE) the size of the circuit C.

An encryption scheme which supports both addition and multiplication (a fully homomorphic scheme) thereby Preserves the ring structure of the plaintext space and is therefore far more powerful. Using such a scheme makes it achievable to let an untrusted party do the computations without ever decrypting the data, and as a result preserving their confidentiality.

An extensively valued application of homomorphic encryption schemes is cloud computing. Currently, the need for cloud computing is growing rapidly, as the data we are dealing out and computing on is getting superior and superior every day.

In order to be clear consider a small example Say, Seeta wants to store a sensitive file m ∈ {0, 1}n on Ram's server. So she sends Ram Encr(m1), ..., Encr(mn). Assume that the file is a database (a catalog of people with specific data about them) and Seeta wants to find out how many of them are 35 years old. Instead of retrieving the data from Ram, decrypting it and searching for the wanted information, she will ask Ram to do the computations, without him knowing what or who he is computing on.

The answer from Ram comes in form of a ciphertext which only she can decrypt with her secret key. The advantage of fully homomorphic encryption has long been acknowledged. The query for constructing such a scheme arises within a year of the improvement of RSA [2].

During this period, the most excellent encryption system was the Boneh-Goh-Nissim cryptosystem [9] which supports estimation of an infinite number of addition operations but one multiplication at the most.

A general reason why a scheme cannot compute circuits of a certain depth is that after a certain amount of computations too much error will build ups, which results the decryption to obtain a wrong value. The decryption usually is able to handle

small amounts of error within a certain range and boots trappable encryption enables "refreshing" after some time. The basic idea of "refreshing" is to encrypt under a first key. Calculate until right before the error grows too large. Encrypt under a second key. Compute the decryption circuit, which since it stopped before the error grew too large, gives the correct value encrypted under the second key. The first key is no longer required. Continue computation under the second key, and repeat the same with a new key as frequently as needed. When the computation has finished, decrypting with the last used key gives the original plaintext.
Gentry's method can be broken down into three main steps:

Step 1: creating an encryption scheme by means of ideal lattices that is somewhat homomorphic, which means it is limited to estimating low-degree polynomials over encrypted data. This scheme is very similar to the Goldreich-Goldwasser- Halevi scheme published in 1997 [10] which is based on lattice problems as well.

Step 2: "Squeezing" the decryption circuit of the original somewhat homomorphic scheme to make it bootstrappable.

Step 3: Bootstrapping to some extent improved original scheme of step 2 to yield the fully homomorphic encryption scheme. This will be done with a "refreshing" procedure.
The innovative idea of Gentry's method of creating a fully homomorphic scheme out of a somewhat homomorphic scheme is the method of squashing and boot-strapping. Mathematically the most appealing step is the first step.

4. SOMEWHAT HOMOMORPHIC SCHEME

The intend of this somewhat homomorphic scheme (SHS) is to build an encryption scheme that is "almost" bootstrappable with respect to a universal set of gates. The first step is to design a SHE scheme which is a scheme that supports some computations over encrypted data. Gentry then demonstrated that if you can handle to design a SHE scheme that supports the evaluation of its own decryption algorithm (and a little more), then there is a common method to transform the SHE scheme into a FHE scheme. A SHE that can estimate its own decryption algorithm homomorphically is called bootstrappable and the procedure that changes a bootstrappable SHE scheme into a FHE scheme is called bootstrapping.

Bootstrapping. First we discuss about how the currently-known SHE schemes work. In general, the ciphertexts of all these schemes contain noise in it and unfortunately this noise gets better as more and more homomorphic operations are carry out. There may be some situations that the encryptions become useless due to much noise i.e., they do not decrypt correctly. This is the main drawback of SHE schemes and this is the reason that they can only carry out a restricted set of computations. Bootstrapping allows us to control this noise.

The design is to take a ciphertext with a huge noise in it and an encryption of the secret key and to homomorphically decrypt the ciphertext. Note that this can only work if the SHE scheme has enough homomorphic ability to evaluate its own decryption algorithm which is why we need the SHE scheme to be bootstrappable. This homomorphically computed decryption will effect in a new encryption of the message but without the noise or at least with less noise than before. More concretely, say we have two ciphertexts:

c1=Epk (m1) and c2=Epk (m2)

with noise n1 and n2, respectively. We can multiply these encryptions using the homomorphic property of the SHE scheme to get an encryption:

c3= Epk (m1 x m2) of m1 x m2 under key pk ,but C3 will now have noise n1xn2. The plan behind bootstrapping is to get rid of this noise as follows. First, we encrypt C3 and sk under pk .This results in two new ciphertexts

C4=Epk(C3) =(Epk (m1 x m2)) and C5= Epk(sk)

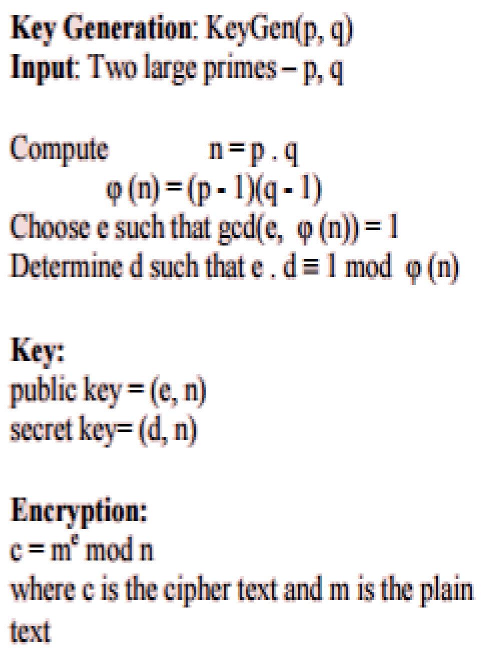
Given C4 and C5, we now homomorphically decrypt C4 using C5. similarly, we compute the following operation over C4 and C5: “decrypt c3= Epk (m1 x m2) using sk“. This is allowed since the scheme has enough homomorphic ability to assess its own decryption algorithm.

Through this technique during a computation whenever the ciphertexts get too noisy, we can remove the main drawback of the SHE scheme and turn it into a FHE scheme.It turns out that constructing a bootstrappable SHE scheme is complex. To do this, Gentry build his scheme using complex methods [1] so a lot of the recent work in FHE has attempted to figure out how to design simpler bootstrappable SHE schemes.

3.1 Partially Homomorphic Cryptosystems

A. RSA-A Multiplicatively Homomorphic Scheme:

In 1978, Rivest, Shamir, and Adleman published their public-key cryptosystem that make use of elementary thoughts from number theory, in their paper "A Method for Obtaining Digital signatures and Public-Key Cryptosystems" [3]. It was one of the first homomorphic cryptosystem. The RSA cryptosystem is the most extensively used public-key cryptosystem. It may be used to give both confidentiality and digital signatures and its security is based on the intractability of the integer factorization problem.



Key Generation: KeyGen(p, q)
Input: Two large primes – p, q

Compute n = p . q
φ (n) = (p - 1)(q - 1)
Choose e such that gcd(e, φ (n)) = 1
Determine d such that e . d ≡ 1 mod φ (n)

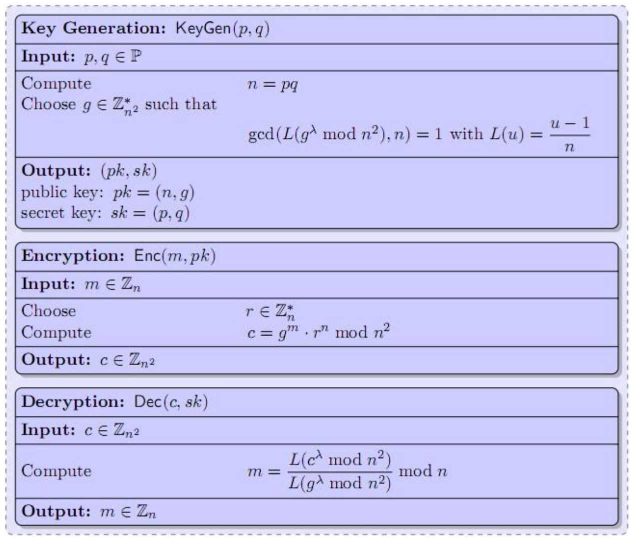
Key:
public key = (e, n)
secret key= (d, n)

Encryption:
$c = m^e \bmod n$
where c is the cipher text and m is the plain text

Fig. 3. RSA Algorithm

The encryption algorithm takes a message m as input from the plaintext space Zn and calculates according ciphertext. c = me mod n. This integer c ∈ Zn cannot be traced back to the original message without the knowledge of p and q, which will be proved later in this section.
Decryption takes as input the ciphertext c and the secret key (d, n) and computes m = cd mod n. Since d is the inverse of e in Zn this is indeed the original message.
The three steps (key generation, encryption and decryption) can be found in the following table.

B. B. Paillier - An Additively Homomorphic Scheme:
Pascal Paillier introduced his cryptosystem in 1999 and published paper "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes" [11]. The proposed technique is based on composite residuosity classes, whose computation is supposed to be computationally difficult. It is a probabilistic asymmetric algorithm for public key cryptography and inherits additive homomorphic properties.
The encryption process takes a message m ∈ Zn as input and randomly chooses an integer r in Z∗ , this random number is used to satisfy the probabilistic algorithm's n property, that one plaintext can have many ciphertexts. It is later revealed that this random variable does not delay the correct decryption, but has the effect of altering the corresponding ciphertext.
The three steps (key generation, encryption and decryption) can be found in the following table:



Key Generation: KeyGen(p,q)	
Input: $p, q \in \mathbb{P}$	
Compute	$n = pq$
Choose $g \in \mathbb{Z}^*_{n^2}$ such that	$\gcd(L(g^\lambda \bmod n^2), n) = 1$ with $L(u) = \frac{u-1}{n}$
Output: (pk, sk)	
public key: $pk = (n, g)$	
secret key: $sk = (p, q)$	

Encryption: Enc(m, pk)	
Input: $m \in \mathbb{Z}_n$	
Choose	$r \in \mathbb{Z}^*_n$
Compute	$c = g^m \cdot r^n \bmod n^2$
Output: $c \in \mathbb{Z}_{n^2}$	

Decryption: Dec(c, sk)	
Input: $c \in \mathbb{Z}_{n^2}$	
Compute	$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$
Output: $m \in \mathbb{Z}_n$	

Fig. 4. Paillier Algorithm

4. PROPOSED FRAMEWORK

The fully homomorphic encryption schemes [1] are very time-consuming. Assuming the evaluation of one gate demanding a refresh, the run-time will be significant as well as the processing of security parameters. A suggestion of a nearly FHE scheme based architecture for allowing the evaluation of any function and producing encrypted data is illustrated in Figure 6. In our proposed architecture, the service provider repartitions the processing among the servers to fasten the evaluation process of any function.

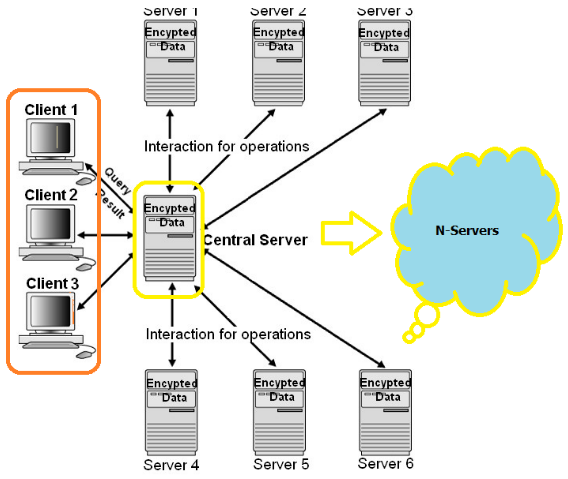


Fig. 5. An architecture of distributed cloud servers for processing encrypted data

In this proposed system, we supply a high leveled architectural scheme during the usage of several servers in the computation. This computational system will nearly allow attaining an FHE, and thus a large number of operations containing additions and multiplications can be performed. For instance, in Fig. 5, it is clearly shown that Client 1 sends a query and requests the results of a given function, Let us consider a function f(x)=ax²+bx+c. In this scenario, the function elements are encrypted and divided into several portions depending on the number of operations (addition and Multiplication), and will be processed independently on N different servers, equivalent to the number of addition operations. At last, the outcome or result is sent back to a Central Server in order to be forwarded to Client 1 and then decrypted.

The advantage is that no longer ciphertext after encryption, unlike the classical method. The keys are simply handled and more security is maintained since it is not possible to read relevant information in distributed systems. In the cloud, the N servers consist of hypervisors hosting multiple virtual machines which support developing the response time and augment the number of the involved computational entities in the distributed system.
In this proposal, we evaluate the added value of the distributed systems in processing operations requested by clients. The scheme of homomorphic encryption is transmitted to the servers and this can be practical and help to develop the security of the cloud in terms of confidentiality of data and performance.

An additional concern that must be measured in our architecture is the confidentiality of the processed data over the distributed systems, nowadays which is the main anxiety of most organizations when using third-party hosting. The approach concerning this issue is the divide the stored data among multiple Cloud service providers to reduce the danger of data violators and increase the parallel processing as well as the number of the servers involved in performing holomorphic encryption. Partitioning and outsourcing the data, applications onto different cloud infrastructures has the advantage of making them uncertain for third-parties and opponents, and thus this assist enhancing the privacy as well as the confidentiality.

Like the stored encrypted data is repartitioned among a Multi-Cloud Architecture belonging to different Cloud Service Providers mentioned in Fig. 6, Client 1 can carry out operations on them and clearly get back the future results. The data is segmented during a Data Partitioning Algorithm (DPA) which permits partitioning, collecting and reconstructing the data. The main operation will be chunked into subsets to be handled by the N Clouds/N Servers. The mixture of N Clouds and homomorphic encryption using N servers gives an improved security strategy which is a safe approach to avoid any potential data breaches even if the data have been previously encrypted.

Selecting a trusted CSP needs a Service Level Agreement (SLA), agreement cooperation and risk estimation. In most cases, it may be logical to believe that a CSP to be trustworthy and handling the clients' sensitive data and applications in a responsible manner.

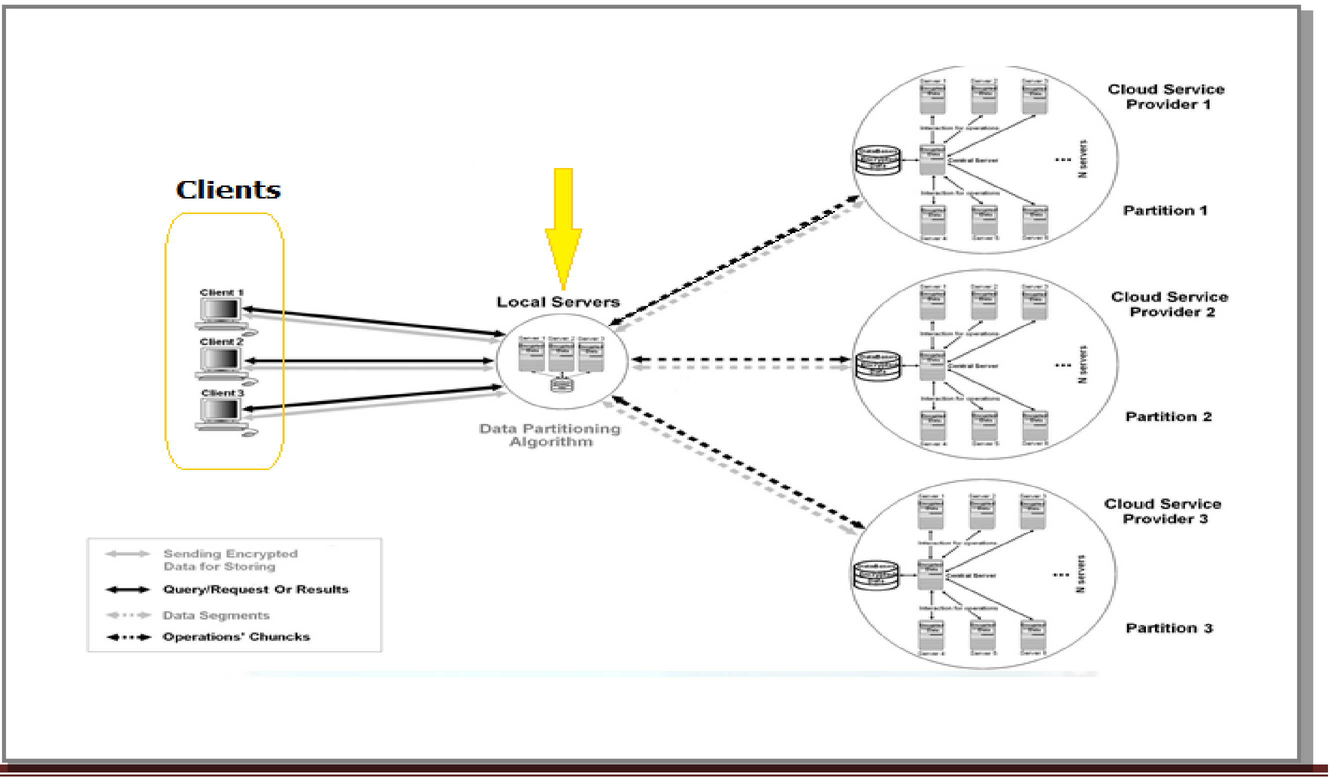


Fig. 6. The suggested architecture to secure data using Homomorphic encryption

5. CONLUSION

In this paper we considering the scheme of Gentry proposed his expansion in regards to FHE, and made colossal endeavor to make FHE more commonsense. While a great deal of advancement has been made, unfortunately we are still while in transit to demonstrate the FHE as practical. Majority of FHE schemes depend on Gentry outline which incorporates of first developing a SHE and afterward utilizing Nobility's bootstrapping method to change over it into a FHE proposal. It changes over out that bootstrapping are a noteworthy bottleneck and that SHE is great requested. In this way, in the event that we consider about functional applications, then it might be sensible to examine what precisely we can do with SHE as an option.

Distributed systems and multi-could models can pass on bunches of focal points to the utilization of homomorphic encryption and making it more reasonable on account of the security of information and applications. The future upgrade will concentrate on the usage of our proposition is to achieve security and execution tests keeping in mind the end goal to clarify its reasonableness.

REFERENCES

[1] C. Gentry, “A fully homomorphic encryption scheme,” Doctoral dissertation, Stanford University, 2009.

[2]K.Lauter, M.Naehrig and V.Vaikunthnathan, “Can homomorphic encryption be practical?”, Proc of 3rd ACM workshop on Cloud Computing Security Workshop , pp 113-124, 2011.

[3] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” Communications of the ACM, 21(2):120-126, 1978.

[4] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” In 18th Annual Eurocrypt Conference (EUROCRYPT'99) Prague, Czech Republic , volume 1592, 1999.

[5] J. Bringe and al., “An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication”, Springer-Verlag, 2007.

[6] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public key cryptosystems,” Communications of the ACM, 21(2):120-126, 1978. Computer Science, pages 223-238.Springer, 1999.

[7] T. ElGamal, “A public key cryptosystem and a signature sche based on discrete logarithms,” IEEE Transactions on Information Theory, 469- 472, 1985.

[8] C. Gentry, “Fully homomorphic encryption using ideal lattices,” InSTOC, Vol. 9, pp. 169-178, 2009.

[9] D. Boneh, E. Goh, and K. Nissim, “Evaluating 2-dnf formulas on ciphertexts,” In Proceedings of Theory of Cryptography (TCC) '05, LNCS 3378, pages 325-341, 2005.

[10] O. Goldreich, S. Goldwasser, and S. Halevi, “Public-key cryptosystems from lattice reduction problems,” In Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology, pages 112-131. Springer-Verlag, 1997.

[11] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” Advances in Cryptology Eurocrypt, 1592:223-238, 1999.

[12] S. Goluch, “The development of homomorphic cryptography: From RSA to Gentry’s privacy homomorphism” Doctoral dissertation, Vienna university of Technology, 2010.